# 2019Android 网络编程总结

## 1. 网络分层

### OSI 七层模型

OSI 七层协议模型主要是：应用层（Application）、表示层（Presentation）、会话层（Session）、传输层（Transport）、网络层（Network）、数据链路层（Data Link）、物理层（Physical）。

## 2. TCP/IP 五层模型

TCP/IP 五层模型：应用层(Application)、传输层(Transport)、网络层(Network)、数据链路层（Data Link）、物理层（Physical）。

## 3. 三次握手与四次挥手

第一次握手：客户端发送 syn 包(syn=j)到服务器，并进入 SYN_SEND 状态，等待服务器确认；
第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；
第三次握手：客户端收到服务器的 SYN + ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP 连接一旦建立，在通信双方中的任何一方主动

关闭连接之前，TCP 连接都将被一直保持下去。断开连接时服务器和客户端均可以主动发起断开 TCP 连接的请求，断开过程需要经过“四次握手”

第一次挥手：客户端发送报文告诉服务器没有数据要发送了
第二次挥手：服务端收到，再发送给客户端告诉它我收到了
第三次挥手：服务端向客户端发送报文，请求关闭连接
第四次挥手：客户端收到关闭连接的请求，向服务端发送报文，服务端关闭连接

## 4. TCP 为什么三次握手不是两次握手，为什么两次握手不安全

为了实现可靠数据传输， TCP 协议的通信双方， 都必须维护一个序列号， 以标识发送出去的数据包中， 哪些是已经被对方收到的。 三次握手的过程即是通信双方相互告知序列号起始值， 并确认对方已经收到了序列号起始值的必经步骤
如果只是两次握手， 至多只有连接发起方的起始序列号能被确认， 另一方选择的序列号则得不到确认

## 5. 为什么 TCP 是可靠的，UDP 早不可靠的?为什么 UDP 比 TCP 快?

TCP/IP 协议拥有三次握手双向机制，这一机制保证校验了数据，保证了他的可靠性。

UDP 就没有了， udp 信息发出后,不验证是否到达对方,所以不可靠。

## 6. http 协议

http 协议是一个基于请求与响应模式的无连接，无状态，应用层的协议，支持 c/s 模式，简单快速，灵活

简单快速：协议简单，通信速度快

灵活：允许传输任意类型的数据对象，由 Content-Type 标记

无连接：每次处理一个请求，处理完成后既断开
无状态：对事务处理没有记忆能力

http 有两种报文：请求报文和响应报文

请求报文由请求行，请求报头，和请求数据组成

请求行：抓包第一行，包括请求方法，url 和 http 版本

请求报头：指的就是题目中“里面的协议头部”

请求数据：指 post 方式提交的表单数据

响应报文由状态行，响应报头，响应正文组成
状态行：状态码
响应报头：同请求报头
响应正文：服务器返回的资源数据

接下来是 http 头部，既请求报头和响应报头，统称消息报头，消息报头可以分为通用报头，请求报头，响应报头，实体报头等

通用报头和实体报头既可以出现在请求报头中，也可以出现在响应报头中，通用报头包含的字段如：Date Connection Cache-Control,实体报头中有

Content-Type Content-Length Content-Language Content-Encoding.
请求报头中包含的字段有：
Host,User-Agent,Accept-Encoding,Accept-Language,Connection
响应报头包含的字段：

Location，Server



## 7. http 的 get 和 post 的区别

http 是应用层的协议，底层基于 TCP/IP 协议，所以本质上，get 和 post 请求都是 TCP 请求。所以二者的区别都是体现在应用层上（HTTP 的规定和浏览器/服务器的限制）

1.参数的传输方式：GET 参数通过 URL 传递，POST 放在 Request body 中。

2.GET 请求在 URL 中传送的参数是有长度限制的，而 POST 没有。

3.对于 GET 方式的请求，浏览器会把 http header 和 data 一并发送出去，服务

器响应 200（返回数据）；而对于 POST，浏览器先发送 header，服务器响应 100 continue，浏览器再发送 data，服务器响应 200 ok（返回数据）。不过要注意，并不是所有浏览器都会在 POST 中发送两次包，比如火狐

4.对参数的数据类型，GET 只接受 ASCII 字符，而 POST 没有限制。

5.GET 比 POST 更不安全，因为参数直接暴露在 URL 上，所以不能用来传递敏感信息。

6.GET 请求只能进行 url 编码，而 POST 支持多种编码方式。

7.GET 在浏览器回退时是无害的，而 POST 会再次提交请求。

8.GET 产生的 URL 地址可以被 Bookmark，而 POST 不可以。

9.GET 请求会被浏览器主动 cache，而 POST 不会，除非手动设置。

## 8. socket 和 http 的区别：

Http 协议：简单的对象访问协议，对应于应用层。Http 协议是基于 TCP 链接的。

tcp 协议：对应于传输层

ip 协议：对应与网络层

TCP/IP 是传输层协议，主要解决数据如何在网络中传输；而 Http 是应用层协议，主要解决如何包装数据。

Socket 是对 TCP/IP 协议的封装，Socket 本身并不是协议，而是一个调用接口（API），通过 Socket，我们才能使用 TCP/IP 协议。

Http 连接：http 连接就是所谓的短连接，及客户端向服务器发送一次请求，服务器端相应后连接即会断掉。

socket 连接：socket 连接及时所谓的长连接，理论上客户端和服务端一旦建立连接，则不会主动断掉；但是由于各种环境因素可能会是连接断开，比如说：服务器端或客户端主机 down 了，网络故障，或者两者之间长时间没有数据传输，网络防火墙可能会断开该链接已释放网络资源。所以当一个 socket 连接中没有数据的传输，那么为了位置连续的连接需要发送心跳消息，具体心跳消息格式是开发者自己定义的。

## 9. TCP 与 UDP 区别总结：

1、TCP 面向连接（如打电话要先拨号建立连接）;UDP 是无连接的，即发送数据之前不需要建立连接

2、TCP 提供可靠的服务。也就是说，通过 TCP 连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP 尽最大努力交付，即不保 证可靠交付

3、TCP 面向字节流，实际上是 TCP 把数据看成一连串无结构的字节流;UDP 是面向报文的

UDP 没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如 IP 电话，实时视频会议等）

4、每一条 TCP 连接只能是点到点的;UDP 支持一对一，一对多，多对一和多对多的交互通信

5、TCP 首部开销 20 字节;UDP 的首部开销小，只有 8 个字节

6、TCP 的逻辑通信信道是全双工的可靠信道，UDP 则是不可靠信道

## 10. https

HTTPS(全称：Hyper Text Transfer Protocol over Secure Socket Layer)，是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。HTTP 是应用层协议，位

于 HTTP 协议之下是传输协议 TCP。TCP 负责传输，HTTP 则定义了数据如何进行包装，在 HTTP 跟 TCP 中间加多了一层加密层 TLS/SSL，SSL 是个加密套件，负责对 HTTP 的数据进行加密。TLS 是 SSL 的升级版。现在提到 HTTPS，加密套件基本指的是 TLS。

传输加密的流程：http 是应用层将数据直接给到 TCP 进行传输，https 是应用层将数据给到 TLS/SSL，将数据加密后，再给到 TCP 进行传输。

HTTPS 是如何加密数据的：

一般来说，加密分为对称加密、非对称加密

**1. 对称加密：**

对称加密的意思就是，加密数据用的密钥，跟解密数据用的密钥是一样的。
对称加密的优点在于加密、解密效率通常比较高。缺点在于，数据发送方、数据接收方需要协商、共享同一把密钥，并确保密钥不泄露给其他人。传输过程中容易被截获。

网上一个很形象的例子：假如现在小客与小服要进行一次私密的对话，他们不希望这次对话内容被其他外人知道。可是，我们平时的数据传输过程中又是明文传输的，万一被某个黑客把他们的对话内容给窃取了，那就难受了。为了解决这个问题，小服这家伙想到了一个方法来加密数据，让黑客看不到具体的内容。该方法是这样子的：在每次数据传输之前，小服会先传输小客一把密钥，然后小服在之后给小客发消息的过程中，会用这把密钥对这些消息进行加密。小客在收到这些消息后，会用之前小服给的那把密钥对这些消息进行解密，这样，小客就能得到密文里面真正的数据了。如果小客要给小服发消息，也同样用这把密钥来对消息进行加密，小服收到后也用这把密钥进行解密。 这样，就保证了数据传输的安全性。

### 2. 非对称加密

非对称加密的意思就是，加密数据用的密钥（公钥），跟解密数据用的密钥（私钥）是不一样的。

网上一个很形象的例子：小服还是挺聪明的，得意了一会之后，小服意识到了密钥会被截取这个问题。倔强的小服又想到了另外一种方法：用非对称加密的方法来加密数据。该方法是这样的：小服和小客都拥有两把钥匙，一把钥匙的公开的（全世界都知道也没关系），称之为公钥；而另一把钥匙是保密（也就是只有自己才知道），称之为私钥。并且，用公钥加密的数据，只有对应的私钥才能解密；用私钥加密的数据，只有对应的公钥才能解密。所以在传输数据的过程中，小服在给小客传输数据的过程中，会用小客给他的公钥进行加密，然后小客收到后，再用自己的私钥进行解密。小客给小服发消息的时候，也一样会用小服给他的公钥进行加密，然后小服再用自己的私钥进行解密。 这样，数据就能安全着到达双方。是什么原因导致非对称加密这种方法的不安全性呢？它和对称加密方法的不安全性不同。非对称加密之所以不安全，是因为小客收到了公钥之后，无法确定这把公钥是否真的是小服。

解决的办法就是数字证书：小服再给小客发公钥的过程中，会把公钥以及小服的个人信息通过 Hash 算法生成消息摘要，为了防止摘要被人调换，小服还会用 CA 提供的私钥对消息摘要进行加密来形成数字签名，当小客拿到这份数字证书之后，就会用 CA 提供的公钥来对数字证书里面的数字签名进行解密得到消息摘要，然后对数字证书里面小服的公钥和个人信息进行 Hash 得到另一份消息摘要，然后把两份消息摘要进行对比，如果一样，则证明这些东西确实是小服的，否则就不是。

## 11. 加密算法

### 1. 对称加密算法

Data Encryption Standard(DES)

DES 是一种典型的块加密方法：将固定长度的明文通过一系列复杂的操作变成同样长度的密文，块的长度为 64 位。同时，DES 使用的密钥来自定义变换过程，因此算法认为只有持有加密所用的密钥的用户才能解密密文。DES 的密钥表面上是 64 位的，实际有效密钥长度为 56 位，其余 8 位可以用于奇偶校验。

DES 现在已经不被视为一种安全的加密算法，主要原因是它使用的 56 位密钥过短。

为了提供实用所需的安全性，可以使用 DES 的派生算法 3DES 来进行加密(虽然 3DES 也存在理论上的攻击方法)。

Advanced Encryption Standard(AES)

AES 在密码学中又称 Rijndael 加密法，用来替代原先的 DES，已经被多方分析且广泛使用。

DES 与 AES 的比较

自 DES 算法公诸于世以来，学术界围绕它的安全性等方面进行了研究并展开了激烈的争论。在技术上，对 DES 的批评主要集中在以下几个方面：

1、作为分组密码，DES 的加密单位仅有 64 位二进制，这对于数据传输来说太小，因为每个分组仅含 8 个字符，而且其中某些位还要用于奇偶校验或其他通讯开销。

2、DES 的密钥的位数太短，只有 56 比特，而且各次迭代中使用的密钥是递推产生的，这种相关必然降低密码体制的安全性，在现有技术下用穷举法寻找密钥已趋于可行。

3、DES 不能对抗差分和线性密码分析。

4、DES 用户实际使用的密钥长度为 56bit，理论上最大加密强度为 256。DES 算法要提高加密强度（例如增加密钥长度），则系统开销呈指数增长。除采用提高

硬件功能和增加并行处理功能外，从算法本身和软件技术方面都无法提高 DES 算法的加密强度。

2. 非对称加密算法

RSA

1977 年由 MIT 的 Ron Rivest、Adi Shamir 和 Leonard Adleman 一起提出，以他们三人姓氏开头字母命名，是一种获得广泛使用的非对称加密算法。

对极大整数做因数分解的难度 (The Factoring Problem) 决定了 RSA 算法的可靠性。换言之，对一个极大整数做因数分解愈困难，RSA 算法就愈可靠。假如有人找到一种快速因数分解的算法的话，那么用 RSA 加密的信息的可靠性就肯定会极度下降。目前看来找到这样的算法的可能性非常小。

DES 与 RSA 的比较

RSA 算法的密钥很长，具有较好的安全性，但加密的计算量很大，加密速度较慢限制了其应用范围。为减少计算量，在传送信息时，常采用传统加密方法与公开密钥加密方法相结合的方式，即信息采用改进的 DES 对话密钥加密，然后使用 RSA 密钥加密对话密钥和信息摘要。对方收到信息后，用不同的密钥解密并可核对信息摘要。

采用 DES 与 RSA 相结合的应用，使它们的优缺点正好互补，即 DES 加密速度快，适合加密较长的报文，可用其加密明文；RSA 加密速度慢，安全性好，应用于 DES 密钥的加密，可解决 DES 密钥分配的问题。

目前这种 RSA 和 DES 结合的方法已成为 EMAIL 保密通信标准。

## 12. Volley

1、 Volley 的特点

Volley 是谷歌大会上推出的网络通信框架（2.3 之前使用 HttpClient，之后使用 HttpUrlConnection），它既可以访问网络获取数据，也可以加载图片，并且在性能方面进行了大幅度的调整，它的设计目的就是适合进行数据量不大但通信频繁的网络操作，而对于大数据量的操作，比如文件下载，表现很糟糕，因为 volley 处理 http 返回的默认实现是 BasicNetwork，它会把返回的流全部导入内存中，下载大文件会发生内存溢出

**2、 Volley 执行的过程：**

默认情况下，Volley 中开启四个网络调度线程和一个缓存调度线程，首先请求会加入缓存队列，，缓存调度线程从缓存队列中取出线程，如果找到该请求的缓存就直接读取该缓存并解析，然后回调给主线程，如果没有找到缓存的响应，则将这个请求加入网络队列，然后网络调度线程会轮询取出网络队列中的请求，发起 http 请求，解析响应并将响应存入缓存，回调给主线程

**3、 Volley 为什么不适合下载上传大文件？为什么适合数据量小的频率高的请求？**

1.volley 基于请求队列，Volley 的网络请求线程池默认大小为 4。意味着可以并发进行 4 个请求，大于 4 个，会排在队列中。并发量小所以适合数据量下频率高的请求

2.因为 Volley 下载文件会将流存入内存中（是一个小于 4k 的缓存池），大文件会导致内存溢出，所以不能下载大文件，不能上传大文件的原因和 1 中差不多，设想你上传了四个大文件，同时占用了 volley 的四个线程，导致其他网络请求都阻塞在队列中，造成反应慢的现象

## 13. OKHttp

**1、 OKHttp 的特点**

1.相较于 Volley，它的最大并发量为 64

2.使用连接池技术，支持 5 个并发的 socket 连接默认 keepAlive 时间为 5 分钟，解决 TCP 握手和挥手的效率问题，减少握手次数

3.支持 Gzip 压缩，且操作对用户透明，可以通过 header 设置，在发起请求的时候自动加入 header，Accept-Encoding: gzip，而我们的服务器返回的时候 header 中有 Content-Encoding: gzip

4.利用响应缓存来避免重复的网络请求

5.很方便的添加拦截器，通常情况下，拦截器用来添加，移除，转换请求和响应的头部信息，比如添加公参等

6.请求失败，自动重连，发生异常时重连，看源码调用 recover 方法重连了一次

7.支持 SPDY 协议(SPDY 是 Google 开发的基于 TCP 的应用层协议，用以最小化网络延迟，提升网络速度，优化用户的网络使用体验。SPDY 并不是一种用于替代 HTTP 的协议，而是对 HTTP 协议的增强。新协议的功能包括数据流的多路复用、请求优先级以及 HTTP 报头压缩。谷歌表示，引入 SPDY 协议后，在实验室测试中页面加载速度比原先快 64%)

8.使用 Okio 来简化数据的访问与存储，提高性能

## 2、 OkHttp 的缺点

1.消息回来需要切到主线程，主线程要自己去写。

2.调用比较复杂，需要自己进行封装。

3.缓存失效：网络请求时一般都会获取手机的一些硬件或网络信息，比如使用的网络环境。同时为了信息传输的安全性，可能还会对请求进行加密。在这些情况下 OkHttp 的缓存系统就会失效了，导致用户在无网络情况下不能访问缓存。

### 3、 OkHttp 框架中都用到了哪些设计模式

1.最明显的 Builder 设计模式，如构建对象 OkHttpClient，还有单利模式

2.工厂方法模式，如源码中的接口 Call

3.观察者模式如 EventListener，监听请求和响应

4.策略模式

5.责任链模式，如拦截器

## 14. Retrofit

Retrofit 底层是基于 OkHttp 实现的，与其他网络框架不同的是，它更多使用运行时注解的方式提供功能

### 1、 原理

通过 java 接口以及注解来描述网络请求，并用动态代理的方式生成网络请求的 request，然后通过 client 调用相应的网络框架（默认 okhttp）去发起网络请求，并将返回的 response 通过 converterFactorty 转换成相应的数据 model, 最后通过 calladapter 转换成其他数据方式（如 rxjava Observable）

### 2、 Retrofit 流程

（1）通过解析 网络请求接口的注解 配置 网络请求参数

（2）通过 动态代理 生成 网络请求对象

（3）通过 网络请求适配器 将 网络请求对象 进行平台适配

（4）通过 网络请求执行器 发送网络请求

（5）通过 数据转换器 解析服务器返回的数据

（6）通过 回调执行器 切换线程（子线程 ->>主线程）

（7）用户在主线程处理返回结果

## 3、 Retrofit 优点

1.可以配置不同 HTTP client 来实现网络请求，如 okhttp、httpclient 等；

2.请求的方法参数注解都可以定制；

3.支持同步、异步和 RxJava；

4.超级解耦；

5.可以配置不同的反序列化工具来解析数据，如 json、xml 等

6.框架使用了很多设计模式